Homework 3 - Loops and Iteration

CS 1301 - Intro to Computing - Spring 2022

Important

- Due Date: **Tuesday, Feburary 1st, 11:59 PM**.
- This is an individual assignment. High-level collaboration is encouraged, **but your submission must be uniquely yours.**
- Resources:
  - TA Helpdesk
  - Email TA's or use class Piazza
  - How to Think Like a Computer Scientist
  - CS 1301 YouTube Channel
- Comment out or delete all function calls. Only import statements, global variables, and comments are okay to be outside of your functions.
- **Read the entire document before starting this assignment.**

The goal of this assignment is to give practice on simple string indexing problems and iteration using for or while loops. The homework will consist of 5 functions for you to implement. You have been given `HW03.py` skeleton file to fill out. Please read this PDF thoroughly as you will find more detailed information to complete your assignment.

Hidden Test Cases: In an effort to encourage debugging and writing robust code, we will be including hidden test cases on Gradescope for some functions. You will not be able to see the input or output to these cases. Below is an example output from a failed hidden test case:

```
Test failed: False is not true
```

Written by Aryan Verma (aryanverma@gatech.edu), Elizabeth Bang (ebang8@gatech.edu)

Average

Function Name: avgTotal()
Parameters: numString (`str`)
Returns: average (`float`)
Description: You want to find the average of integers present inside a string. Given a string that contains only numbers, return the average value of the numbers that are in the string.

Note: If numString is an empty string, the function should return the average as `0.0`.

```
>>> avgTotal("3579")
6.0
```

```
>>> avgTotal("")
0.0
```

Safe Decoder

Function Name: safeDecoder()
Parameters: characterString (`str`)
Returns: passcodeString (`str`)
Description: You are given an encrypted string with numbers, letters, and characters. You want to extract the indices of the numbers in the string to find the passcode to a safe. Given the characterString, write a function that returns a new string which is made up of the indices of each digit found in the original string. The order of the returned indices should match the order in which the original digits of the characterString appeared.

```
>>> safeDecoder("ab*n3a&ca4o21$")
'491112'
```

```
>>> safeDecoder("0p3n_Up_th3_$4f3!")
'02101315'
```

Test Score

Function Name: testScore()
Parameters: test1(`str`), test2(`str`)
Returns: maxPercentage (`float`)
Description: You just got back the results of the two tests that you took last week and you need to compare the percentage scored on the two tests. Each test had 5 questions with 5 being the maximum amount of points for each question. You are given two strings which include the points obtained on each individual question of a test seperated by a `"_"`. You need to return a float which provides the highest percentage scored out of the two tests you took. **Assume that you can only recieve a whole number between 0-5 on each question of a test.**

Note: If the percentage of both the tests is the same return a string `"Same Percentage"`.

```
>>> testScore("5_5_5_5_5", "4_4_4_4_4")
100.0
```

```
>>> testScore("3_5_4_2_5", "2_5_5_2_5")
'Same Percentage'
```

Trip

Function Name: trip()
Parameters: tripTotalCost (`float`), bankBalance(`float`), interestRate (`float`)
Returns: number of months (`int`)
Description: You want to figure out how long it will take for you to save up and go on the trip of your dreams. Thankfully, you've got some money in the bank that's growing every month with compound interest. Given the total cost of your trip(`tripTotalCost`), the money you currently have in the bank(`bankBalance`), and the interest rate of your bank account **given as a percent**(`interestRate`), return how many months it will take for you to have enough money for your trip.

Note: You may assume that all parameters will be positive numbers. You do **not** need to use the the compound interest formula for this function.

```
>>> trip(50000.0, 10000.0, 5.0)
33
```

```
>>> trip(100000.0, 112.15, 2.1)
327
```

Right Triangles

Function Name: rightTriangles()
Parameters: numRows (`int`), character (`str`)
Returns: None
Description: You find that if you start with one character and for every row you increase the number of characters by 1, you can print a right triangle. Write a function that takes in a number of rows and a character and print a right triangle with the a height of the given number of rows that is made up of the given character. If you cannot make a right triangle print the string `"No Triangle"`.

Note: You need at least two rows to make a right triangle.

```
>>> rightTriangles(4, '*')
'*'
'**'
'***'
'****'
```

```
>>> rightTriangles(2, '!')
'!'
'!!'
```

Grading Rubric

Function	Points
avgTotal()	20
safeDecoder()	20
testScore()	20
trip()	20
rightTriangles()	20
Total	**100**

Provided

The `HW03.py` skeleton file has been provided to you. This is the file you will edit and implement. All instructions for what the functions should do are in this skeleton and this document.

Submission Process

For this homework, we will be using Gradescope for submissions and automatic grading. When you submit your `HW03.py` file to the appropriate assignment on Gradescope, the autograder will run automatically. The grade you see on Gradescope will be the grade you get, unless your grading TA sees signs of you trying to defeat the system in your code. You can re-submit this assignment an unlimited number of times until the deadline; just click the “Resubmit” button at the lower right-hand corner of Gradescope. You do not need to submit your `HW03.py` on Canvas.